Science of Computer Programming 3 (1983) 217–222
North Holland

SMALL PROGRAMMING EXERCISES

M. REM

Department of Mathematics and Computing Science, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

A beautiful solution to a programming problem can be fascinating and exciting. This is a fortiori so when the solution is the embodiment of a convincing derivation from the problem specification. Over the years a collection of such problems and solutions have been constructed at Eindhoven University of Technology. They were used to test the programming abilities of the students. At other universities similar collections must have been compiled. Thus, a growing repertoire of beautiful solutions is emerging. In this new section I want to share with the readers of Science of Computer Programming the fascination of arriving at—or being exposed to—such solutions to programming exercises.

This is not a 'traditional' problem section with readers submitting solutions and an editor keeping score of the correct solutions. Our exercises are just too simple for that: an experienced programmer will usually find a satisfactory solution in a few minutes. That is why I call them 'exercises'. Many solutions I present may, nevertheless, be surprising. They may be simpler than the reader envisaged, or just clearer, or they may involve an unexpected correctness argument. Of course, I may occasionally revert to disappointing or questionable solutions. In such cases I would like my readers to criticize me and to propose improvements or alternative solutions. In this way we may foster the science of programming while enjoying the programming exercises. It goes without saying that I also welcome suggestions for new exercises.

Exercises in textbooks are often either elementary or very complicated. This section aims at a level in-between. Most exercises will require some thought before they can be solved. They are of the type one could assign to a student taking a modern programming course, say one based on David Gries's text *The Science of Programming* (Springer, Berlin, 1981).

This first time I offer three exercises. In order for the reader to become acquainted with my style of presenting exercises and solutions, I include one of the solutions. The solutions to the other two will appear in the next issue. Exercise 0 I owe to W.H.J. Feijen. Exercise 1 is a mild variation of a programming problem received from Rutger M. Dijkstra. Exercise 2 was composed by Jan L.A. van de Snepscheut. The notation in which I phrase the exercises and the solutions—i.e., the notation for functional specifications and programs—I owe to Edsger W. Dijkstra.

Notational interlude. The expression

$$(\mathbf{N}\, i,j\colon D(i,j)\colon P(i,j))$$

denotes the number of solutions of $P(i,j)$ for all pairs (i,j) satisfying $D(i,j)$.

$$(\mathbf{A}\, i,j\colon D(i,j)\colon P(i,j)) \equiv (\mathbf{N}\, i,j\colon D(i,j)\colon \neg P(i,j)) = 0,$$

$$(\mathbf{E}\, i,j\colon D(i,j)\colon P(i,j)) \equiv \neg(\mathbf{A}\, i,j\colon D(i,j)\colon \neg P(i,j)).$$

Likewise,

$$(\mathbf{MAX}\, i,j\colon D(i,j)\colon E(i,j))$$

denotes the maximum value of $E(i,j)$ for all pairs (i,j) satisfying $D(i,j)$.

$$(\mathbf{MIN}\, i,j\colon D(i,j)\colon E(i,j)) = -(\mathbf{MAX}\, i,j\colon D(i,j)\colon -E(i,j)).$$

We also employ infix-operators **max** and **min**: b **max** c denotes the maximum of b and c.

$$(\mathbf{S}\, i,j\colon D(i,j)\colon E(i,j))$$

denotes the sum of the values $E(i,j)$ for all pairs (i,j) satisfying $D(i,j)$.

For $N \geq 0$

$$A(i\colon 0 \leq i < N)$$

denotes the sequence $A(0), A(1), \ldots, A(N-1)$ of N elements. This notation is used for (one-dimensional) arrays and for segments (contiguous subsequences) thereof. $A(i\colon 0 \leq i < N)$ is called

ascending if $(\mathbf{A}\, i\colon 0 < i < N\colon A(i-1) \leq A(i))$,

descending if $(\mathbf{A}\, i\colon 0 < i < N\colon A(i-1) \geq A(i))$,

increasing if $(\mathbf{A}\, i\colon 0 < i < N\colon A(i-1) < A(i))$,

decreasing if $(\mathbf{A}\, i\colon 0 < i < N\colon A(i-1) > A(i))$.

Exercise 0: longest almost ascending segment

For an integer array $A(i\colon 0 \leq i < N)$ an 'almost ascending segment' of length $q-p$ is a segment $A(i\colon p \leq i < q)$, $0 \leq p < q \leq N$, for which there exists at most one value i in the range $p < i < q$ satisfying $A(i-1) > A(i)$. Requested is a program for determining the maximum length among the almost ascending segments.

With $AAS(p,q)$ denoting

$$(\mathbf{N}\, i\colon p < i < q\colon .A(i-1) > A(i)) \leq 1$$

this exercise may be phrased as follows.

Find a statement list S such that

$|[N: int\{N \geq 1\}$
; $A(i: 0 \leq i < N)$: **array of** *int*
; $|[c: int$
; S
$\{c = (\mathbf{MAX}\ p,q: 0 \leq p < q \leq N \wedge AAS(p,q): q - p)\}$
$]|$
$]|$

The above is a functional specification. It consists of an outer block and an inner block, each delineated by a bracket pair "$|[\ldots]|$". The variables declared in the outer block of a functional specification are to be left unchanged by S. They constitute the values (the 'input values') S has to operate upon. Relevant properties of their values, $N \geq 1$ in this case, are stated between curly brackets next to the declarations. The conjunction of these properties is known as the precondition of S. The inner block of a functional specification contains the declarations of the variables that are to be assigned a value (an 'output value') by S. Initially these variables have undefined values. The statement list S should be chosen in such a way that the final values of the variables of the inner block, just c in this case, satisfy the property stated between curly brackets at the end of the inner block. This property is known as the postcondition of S. Thus, given its precondition, S should establish its post-condition by assigning appropriate values to the variables of the inner block.

Solution of Exercise 0

We derive from the postcondition an invariant $P0$ by replacing the constant N by a suitably bounded variable n:

$P0$: $\quad c = (\mathbf{MAX}\ p,q: 0 \leq p < q \leq n \wedge AAS(p,q): q - p) \wedge 1 \leq n \leq N.$

This gives rise to a program with the following S:

S: $\quad |[\ n: int$
; $c,n := 1,1\ \{P0\}$
; **do** $n \neq N$
$\rightarrow \{P0 \wedge n \neq N\}$
"establish $P0^n_{n+1}$"
; $n := n + 1\ \{P0\}$
od $\{P0 \wedge n = N\}$
$]|$

Annotations are again given between curly brackets. $P0^n_{n+1}$ denotes the predicate $P0$ in which $n+1$ is substituted for n. The repetition in S terminates, since the bound function $N-n$, which by virtue of $P0$ is positive, is decreased per step of the repetition. The part of S that still needs to be coded is written between quotation marks. We focus our attention on establishing $P0^n_{n+1}$.

$P0^n_{n+1}$:

$$c = (\mathbf{MAX}\ p,q: 0 \leq p < q \leq n+1 \wedge AAS(p,q): q-p)$$
$$\wedge\ 1 \leq n+1 \leq N.$$

The latter conjunct is implied by $P0 \wedge n \neq N$. The first conjunct may be written as

$$c = (\mathbf{MAX}\ p,q: 0 \leq p < q \leq n+1 \wedge AAS(p,q): q-p)$$
$$\mathbf{max}(\mathbf{MAX}\ p: 0 \leq p < n+1 \wedge AAS(p,n+1): n+1-p).$$

This is equivalent to

$$c = (\mathbf{MAX}\ p,q: 0 \leq p < q \leq n \wedge AAS(p,q): q-p)$$
$$\mathbf{max}(n+1-(\mathbf{MIN}\ p: 0 \leq p < n+1 \wedge AAS(p,n+1): p)).$$

The above formula suggests to extend the invariant $P0$ to a new invariant $P0 \wedge P1$.

$P1$: $d = (\mathbf{MIN}\ p: 0 \leq p < n \wedge AAS(p,n): p)$.

This yields

S: $[\![\ n,d: int$
; $c,n,d := 1,1,0\ \{P0 \wedge P1\}$
; **do** $n \neq N$
$\rightarrow \{P0 \wedge P1 \wedge n \neq N\}$
"establish $P1^n_{n+1}$"
; $c := c\ \mathbf{max}(n+1-d)$
; $n := n+1\{P0 \wedge P1\}$
od
$]\!]$

We are left with establishing $P1^n_{n+1}$.

$P1^n_{n+1}$:

$$d = (\mathbf{MIN}\ p: 0 \leq p < n+1 \wedge AAS(p,n+1): p)$$

For $n \geq 1$, which is guaranteed by $P0$, $P1^n_{n+1}$ is equivalent to

$$d = \begin{cases} (\mathbf{MIN}\, p: 0 \leq p < n \wedge AAS(p,n): p) & \text{if } A(n-1) \leq A(n), \\ (\mathbf{MIN}\, p: 0 \leq p < n \wedge AS(p,n): p) & \text{if } A(n-1) > A(n) \end{cases}$$

in which $AS(p, n)$ denotes

$$(\mathbf{N}i: p < i < n: A(i-1) > A(i)) = 0.$$

This suggests another extension to the invariant, yielding $P0 \wedge P1 \wedge P2$ as the new invariant.

$P2$: $\quad e = (\mathbf{MIN}\, p: 0 \leq p < n \wedge AS(p,n): p)$.

We thus obtain as our solution

```
S:  |[ n,d,e: int
    ; n,c,d,e := 1,1,0,0 {P0 ∧ P1 ∧ P2}
    ; do n ≠ N
        → if A(n-1) ≤ A(n) → skip
          □ A(n-1) > A(n) → d,e := e,n
          fi {P1^n_{n+1} ∧ P2^n_{n+1}}
      ; c := c max(n+1-d)
      ; n := n+1 {P0 ∧ P1 ∧ P2}
      od
    ]|
```

Exercise 1: longest smooth segment

For an integer array $A(i: 0 \leq i < N)$ a 'smooth segment' of length $q - p$ is a segment $A(i: p \leq i < q)$ satisfying

$$0 \leq p < q \leq N \wedge SS(p, q)$$

in which $SS(p, q)$ denotes

$$(\mathbf{A}i,j: p \leq i < q \wedge p \leq j < q: A(i) - A(j) \leq 1).$$

Find a statement list S such that

```
|[N: int{N ≥ 1}
; A(i: 0 ≤ i < N): array of int
; |[ c: int
  ; S
```

$\{c = (\mathbf{MAX}\, p, q: 0 \leq p < q \leq N \wedge SS(p,q): q - p)\}$

]||

]|

Exercise 2: Fibolucci

For given integer values X and Y a function f on the natural numbers is defined by

$f(0) = 1$

$f(1) = 1,$

$f(i+2) = X \cdot f(i) + Y \cdot f(i+1) \quad \text{for } i \geq 0.$

Find a statement list S (not containing an auxiliary array) such that

|[N, X, Y: *int* $\{N \geq 1\}$

; |[a: *int*

; S

$\{a = (\mathbf{S}i: 0 \leq i \leq N: f(i) \cdot f(N-i))\}$

]||

]|